UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/698,820	10/31/2003	Matthew Englehart	MWS-062	1288

959 7590 09/12/2007
LAHIVE & COCKFIELD, LLP
ONE POST OFFICE SQUARE
BOSTON, MA 02109-2127

EXAMINER	
CHEN, QING	
ART UNIT	PAPER NUMBER
2191	
MAIL DATE	DELIVERY MODE
09/12/2007	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Office Action Summary	**Application No.** 10/698,820	**Applicant(s)** ENGLEHART ET AL.	
	Examiner Qing Chen	**Art Unit** 2191	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *24 July 2007*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-26* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-26* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on *24 July 2007* is/are: a)☒ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)

2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)

3) ☐ Information Disclosure Statement(s) (PTO/SB/08)
Paper No(s)/Mail Date _____.

4) ☐ Interview Summary (PTO-413)
Paper No(s)/Mail Date. _____ .

5) ☐ Notice of Informal Patent Application

6) ☐ Other: _____.

DETAILED ACTION

1. This Office action is in response to the RCE filed on July 24, 2007.

2. **Claims 1-26** are pending.

3. **Claims 1, 12, and 16-26** have been amended.

4. The objection to the drawings is withdrawn in view of Applicant's amendments to the drawings.

5. The objections to Claims 16-26 are withdrawn in view of Applicant's amendments to the claims.

6. The 35 U.S.C. § 101 rejections of Claims 16-26 are withdrawn in view of Applicant's amendments to the claims.

Response to Amendment

Claim Rejections - 35 USC § 102

7. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless –
>
> (b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

8. **Claims 1-4, 6-8, 12-14, 16-19, and 21-23** are rejected under 35 U.S.C. 102(b) as being anticipated by **Cheng et al. (US 2002/0010908)**.

As per **Claim 1**, Cheng et al. disclose:

- providing a user interface with a plurality of selectable parameters for a custom storage class, said custom storage class specifying the manner in which an automatic code generator creates source code that implements functionality of said graphical model, including source code corresponding to data referenced by said graphical model in said graphical modeling and execution environment *(see Figures 4, 6, and 7; Paragraph [0023], "FIG. 4 shows an exemplary command graphical user interface ("GUI") 200 for command structure manifest 110 described with respect to FIG. 2. Command structure manifest 110 enables a developer to visually manipulate the command structure by adding and deleting command nodes at any level." and "... GUT (sic) 200 also shows parameters and handler functions associated with each command node."; Paragraph [0026], "FIG. 7 shows an exemplary GUI 400 for command node editor 120."; Paragraph [0028], "The entering of parameters is also accomplished via GUI 400 by adding the desired parameters to parameter field 410."; Paragraph [0043], "Handler code generation engine 135 automatically generates this software code using the information entered by the developer and the parameter and handler function definitions generated by command structure generation engine 145.")*; and

- creating a custom storage class in said graphical modeling and execution environment utilizing parameters selected by a user from said plurality of selectable parameters *(see Figure 6: 360; Paragraph [0039], "... the handler function definitions and parameter definitions are generated by command structure generation engine." and "... command structure generation engine takes the information input by the developer and generates a file containing the information for the handler functions and parameters."; Paragraph [0040], "This code describes an exemplary parameter definition array mCommand3Params for command3.";*

Paragraph [0042], "This code describes an exemplary handler function definition array mCommand3Handlers for command3.").

As per **Claim 2**, the rejection of **Claim 1** is incorporated; and Cheng et al. further disclose:

- providing a view of salient aspects of the source code generated by said automatic code generator utilizing the user-selected parameters *(see Figure 11; Paragraph [0046], "GUI displays the code generated by handler code generation engine so that the developer may view, review and accept the automatically generated code.").*

As per **Claim 3**, the rejection of **Claim 2** is incorporated; and Cheng et al. further disclose:

- changing the user-selected parameters for said custom storage class in said user interface *(see Paragraph [0051], "... a developer edits parameters in a handler function through GUI ...")*; and

- adjusting the source code generated by said automatic code generator to reflect the change in user-selected parameters *(see Paragraph [0051], "... the command structure, the handler function definitions, the parameter definitions and the handler function code is automatically generated based on the information provided by the developer and therefore may need to be revised based on any changed or additional information provided by the developer." and "... these changes will be automatically reconciled in the handler function code by handler code generation engine.").*

As per **Claim 4**, the rejection of **Claim 3** is incorporated; and Cheng et al. further disclose:

- displaying salient aspects of the adjusted source code in said view of salient aspects of the source code *(see Paragraph [0044], "This code may be viewed as it is being generated in code view field of GUI as parameters are being added to the handler function.").*

As per **Claim 6**, the rejection of **Claim 1** is incorporated; and Cheng et al. further disclose:

- wherein said custom storage class declares macros for instances of constant data *(see Paragraph [0038], "... #define kCommand3Help "\help string for command 3\" ... ").*

As per **Claim 7**, the rejection of **Claim 1** is incorporated; and Cheng et al. further disclose:

- wherein said custom storage class declares variables for instances of constant data *(see Paragraph [0040], "The exemplary code may include the following information: keyword or name, data type (e.g., integer, boolean, etc.), a unique bitmask identifier ... ").*

As per **Claim 8**, the rejection of **Claim 1** is incorporated; and Cheng et al. further disclose:

- wherein said user-selected parameters control at least one of the manner in which automatically generated source code is defined, declared, accessed and addressed *(see*

Paragraph [0043], "Handler code generation engine automatically generates this software code using the information entered by the developer and the parameter and handler function definitions generated by command structure generation engine.").

As per **Claim 12**, Cheng et al. disclose:

- a display device *(see Figure 8)* for:

  - displaying a user interface with a plurality of selectable parameters for a custom storage class, said custom storage class specifying the manner in which an automatic code generator creates source code that implements functionality of said graphical model *(see Figures 4, 6, and 7; Paragraph [0023], "FIG. 4 shows an exemplary command graphical user interface ("GUI") 200 for command structure manifest 110 described with respect to FIG. 2. Command structure manifest 110 enables a developer to visually manipulate the command structure by adding and deleting command nodes at any level." and "... GUT (sic) 200 also shows parameters and handler functions associated with each command node."; Paragraph [0026], "FIG. 7 shows an exemplary GUI 400 for command node editor 120."; Paragraph [0028], "The entering of parameters is also accomplished via GUI 400 by adding the desired parameters to parameter field 410."; Paragraph [0043], "Handler code generation engine 135 automatically generates this software code using the information entered by the developer and the parameter and handler function definitions generated by command structure generation engine 145.")*;

  - displaying a view of salient aspects of the source code generated by said automatic code generator utilizing the user-selected parameters *(see Figure 11; Paragraph*

[0046], "GUI displays the code generated by handler code generation engine so that the developer may view, review and accept the automatically generated code."); and

- a processor for creating a custom storage class in said graphical modeling and execution environment, said custom storage class created utilizing parameters selected by a user from said plurality of selectable parameters *(see Figure 6: 360; Paragraph [0039], "... the handler function definitions and parameter definitions are generated by command structure generation engine." and "... command structure generation engine takes the information input by the developer and generates a file containing the information for the handler functions and parameters."; Paragraph [0040], "This code describes an exemplary parameter definition array mCommand3Params for command3."; Paragraph [0042], "This code describes an exemplary handler function definition array mCommand3Handlers for command3.")*.

As per **Claim 13**, the rejection of **Claim 12** is incorporated; and Cheng et al. further disclose:

- wherein the user-selected parameters for said custom storage class in said user interface are changed and the source code generated by said automatic code generator is adjusted to reflect the change user-selected parameters *(see Paragraph [0051], "... a developer edits parameters in a handler function through GUI ..." and "... the command structure, the handler function definitions, the parameter definitions and the handler function code is automatically generated based on the information provided by the developer and therefore may need to be revised based on any changed or additional information provided by the developer." and "...*

these changes will be automatically reconciled in the handler function code by handler code generation engine.").

As per **Claim 14**, the rejection of **Claim 13** is incorporated; and Cheng et al. further disclose:

- wherein the adjusted source code is displayed in said view of salient aspects of the source code *(see Paragraph [0044], "This code may be viewed as it is being generated in code view field of GUI as parameters are being added to the handler function.").*

Claims 16-19 and 21-23 are computer-readable medium claims corresponding to the method claims above (Claims 1-4 and 6-8, respectively) and, therefore, are rejected for the same reasons set forth in the rejections of Claims 1-4 and 6-8, respectively.

Claim Rejections - 35 USC § 103

9. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

10. **Claims 5, 15, and 20** are rejected under 35 U.S.C. 103(a) as being unpatentable over **Cheng et al. (US 2002/0010908)** in view of **Childress et al. (US 2004/0085357).**

As per **Claim 5**, the rejection of **Claim 2** is incorporated; however, Cheng et al. do not disclose:

- wherein said view of salient aspects of the source code automatically generated includes at least one token, said token being symbolically representative of a non-displayed segment of source code.

Childress et al. disclose:

- wherein said view of salient aspects of the source code automatically generated includes at least one token, said token being symbolically representative of a non-displayed segment of source code *(see Paragraph [0115], "... code may be included as 'hidden' text in one or more sections of documents, and may be used in constructing header tables and text tables.")*.

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Childress et al. into the teaching of Cheng et al. to include wherein said view of salient aspects of the source code automatically generated includes at least one token, said token being symbolically representative of a non-displayed segment of source code. The modification would be obvious because one of ordinary skill in the art would be motivated to minimize the usage of available memory.

As per **Claim 15**, the rejection of **Claim 12** is incorporated; however, Cheng et al. do not disclose:

- wherein said view of salient aspects of the source code includes at least one token, said token being symbolically representative of a non-displayed segment of code.

Childress et al. disclose:

- wherein said view of salient aspects of the source code includes at least one token, said token being symbolically representative of a non-displayed segment of code *(see Paragraph [0115], "... code may be included as 'hidden' text in one or more sections of documents, and may be used in constructing header tables and text tables.").*

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Childress et al. into the teaching of Cheng et al. to include wherein said view of salient aspects of the source code includes at least one token, said token being symbolically representative of a non-displayed segment of code. The modification would be obvious because one of ordinary skill in the art would be motivated to minimize the usage of available memory.

Claim 20 is rejected for the same reason set forth in the rejection of Claim 5.

11. **Claims 9, 10, 24, and 25** are rejected under 35 U.S.C. 103(a) as being unpatentable over **Cheng et al. (US 2002/0010908)** in view of **Davidov et al. (US 2003/0225774)**.

As per **Claim 9**, the rejection of **Claim 1** is incorporated; however, Cheng et al. do not disclose:

- wherein said user-selected parameter includes a non-portable directive to a compiler.

Davidov et al. disclose:

- wherein said user-selected parameter includes a non-portable directive to a compiler *(see Paragraphs [0092] and [0214], "Command line options or directives for the compiler ..." and "The compiler compiles Java source code produced by the generator according to supplied directives.")*.

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Davidov et al. into the teaching of Cheng et al. to include wherein said user-selected parameter includes a non-portable directive to a compiler. The modification would be obvious because one of ordinary skill in the art would be motivated to conveniently and dynamically create software programs that can be executed on a computer system.

As per **Claim 10**, the rejection of **Claim 9** is incorporated; however, Cheng et al. do not disclose:

- wherein said non-portable directive to a compiler assigns data to at least one memory location in said electronic device.

Davidov et al. disclose:

- wherein said non-portable directive to a compiler assigns data to at least one memory location in said electronic device *(see Paragraph [0160], "The data is loaded when the application is started, and is saved when the application is destroyed. This type of persistence uses the device records management system (RMS), for example, non-volatile memory.")*.

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Davidov et al. into the teaching of Cheng et al.

to include wherein said non-portable directive to a compiler assigns data to at least one memory location in said electronic device. The modification would be obvious because one of ordinary skill in the art would be motivated to store data that can be utilized at a later time.

Claim 24 is rejected for the same reason set forth in the rejection of Claim 9.

Claim 25 is rejected for the same reason set forth in the rejection of Claim 10.

12. **Claims 11 and 26** are rejected under 35 U.S.C. 103(a) as being unpatentable over **Cheng et al.** **(US 2002/0010908)** in view of **DeMaster** **(US 6,066,181)**.

As per **Claim 11,** the rejection of **Claim 1** is incorporated; however, Cheng et al. do not disclose:

- creating a separate header file with said automatic code generator in response to the selection of one of said plurality of user-selected parameters.

DeMaster discloses:

- creating a separate header file with said automatic code generator in response to the selection of one of said plurality of user-selected parameters *(see Column 4: 55-57, "... the Java native interface code generator generates Java Classes and data conversion code stubs (and related header files).")*.

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of DeMaster into the teaching of Cheng et al. to include creating a separate header file with said automatic code generator in response to the selection of one of said plurality of user-selected parameters. The modification would be obvious because one of ordinary skill in the art would be motivated to allow software portability, so that software applications may easily be moved to another environment *(see DeMaster – Column 1: 23-25)*.

Claim 26 is rejected for the same reason set forth in the rejection of Claim 11.

Response to Arguments

13. Applicant's arguments filed on July 24, 2007 have been fully considered, but they are not persuasive.

In the remarks, Applicant argues that:

a) Applicants respectfully submit that Cheng fails to disclose "custom storage class specifying the manner in which an automatic code generator creates source code that implements functionality of said graphical model, including source code corresponding to data referenced by said graphical model in said graphical modeling and execution environment," as recited in claim 1, because Cheng does not address storage classes. The Examiner points to the handler function definitions and parameter definitions as disclosing the custom storage class recited in claim 1 (Office Action, paragraph 12). Applicants respectfully disagree with the Examiner's characterization of Cheng. Applicants contend that the handler function definitions and parameter definitions, discussed in Cheng, are not synonymous with the custom storage class recited in claim 1, as set forth below.

As discussed in Applicants' Specification at pages 1-2, each item of data in a graphical model is defined to have a data storage class. Data is represented in software source code produced from the graphical model in a manner that is prescribed by its data storage class (Specification, page 1). The software source code references data in a number of different ways including defining data, declaring data, initializing data, reading a value of data, assigning the value of data, and the choice o f storage class controls how each o f these references are generated (Specification, pages 1-2).

As discussed in Applicants' Specification at page 2, code generators may provide predefined sets of storage classes, and they may also permit the user to define new, custom storage classes with user-defined characteristics. Changes to the unique set of instructions defining a custom storage class collectively apply to the set of data of that class (Specification, page 2). Common software engineering practices that may be enabled with custom storage classes include, but are not limited to, embedding a data item in a bit field, embedding a data item in a structure, embedding a data item in a union, using platform-specific declarations in the data declaration, defining the scope and storage of the data, declaring data using arbitrary C types, and accessing data through function calls (Specification, page 2).

As discussed above, Cheng discusses that execution of the handler function code associated with the handler function of a command node causes the operating system to carry out the command entered by the user (Cheng, paragraph 18). The handler function code is generated by a handler code generation engine 135 which uses information entered by the developer and parameter and handler function definitions (Cheng, paragraph 43). The parameter definitions and handier function definitions provide information on how the commands typed in the command-line interface can be carried out by the operating system. Cheng does not disclose that the parameter definitions or the handler function definitions specify the manner in which handler function code is generated corresponding to data referenced by said graphical model in said graphical modeling and execution environment. In fact, Cheng does not mention referencing of data by a graphical model. In contrast, claim 1 requires a custom storage class, with the "custom storage class specifying the manner in which an automatic code generator creates source code that implements functionality of said graphical model, including source code corresponding to data referenced by said graphical model in said graphical modeling and execution environment." For example, such source code may include instructions on embedding a data item in a bit field, embedding a data item in a structure, embedding a data item in a union, etc, as described above (Specification, page 2).

Examiner's response:

a) Examiner disagrees with Applicant's assertion that the handler function definitions and the parameter definitions are not synonymous with the custom storage class recited in Claim 1. Note that the handler function definitions and the parameter definitions are both configured by the developer and, therefore, are customizable *(see Figures 8 and 9; Paragraph [0029], "FIG. 8 shows an exemplary GUI 450 for parameter editor 140."; Paragraph [0032], "FIG. 9 shows an exemplary GUI 500 for handler editor 130.").*

Furthermore, the handler function definitions and the parameter definitions specify the manner in which source code is generated that implements functionality of the graphical model *(see Paragraph [0043], "Handler code generation engine 135 automatically generates this* *software code using the information entered by the developer and the parameter and handler function definitions generated by command structure generation engine 145. ").* Thus, the handler function definitions and the parameter definitions both specify the manner in which the handler code generation engine creates software code.

Also, note that the handler function code is generated as a result of a new command node being inserted into the command structure *(see Figure 4; Paragraph [0023], "FIG. 4 shows an exemplary command graphical user interface ("GUI") 200 for command structure manifest 110 described with respect to FIG. 2. Command structure manifest 110 enables a developer to visually manipulate the command structure by adding and deleting command nodes at any level. " and "... GUT (sic) 200 also shows parameters and handler functions associated with each command node. ").* Thus, the handler function code is generated corresponding to command nodes (data) referenced by the command structure (graphical model).

In addition, although the claims are interpreted in light of the specification, limitations from the specification are not read into the claims. See *In re Van Geuns*, 988 F.2d 1181, 26 USPQ2d 1057 (Fed. Cir. 1993).

In the remarks, Applicant argues that:

b) Applicants respectfully submit that Cheng fails to disclose "source code that implements functionality of said graphical model," as recited in claim 1. It appears from the Examiner's remarks at paragraph 17 of the office action that the Examiner is pointing to the software code generated by the handler code generation engine in Cheng as disclosing the source code recited in claim 1. As recited in claim 1, execution of the source code executes the graphical model.

Cheng does not disclose that execution of the code generated by the handler code generation engine executes a graphical model, as required by claim 1.

The command tree and the CLI of Cheng are not graphical models or graphical modeling and execution environments, as required by claim 1. Cheng teaches that executing the software code associated with a handler function causes the operating system to carry out the particular command typed by the user in the command-line interface (Cheng, paragraph 22). Thus, when a user enters a command at the interface, the operating system traverses the command tree branch and reaches an appropriate node (Cheng, paragraph 22). At the node, the operating system retrieves the appropriate handler function and executes the software code associated with the handler function (Cheng, paragraph 22). This results in the operating system carrying out the command entered by the user (Cheng, paragraph 22). In contrast, claim 1 requires that execution of the source code implements the functionality of the graphical model. Execution of a command entered by a user at a command-line interface, as discussed in Cheng, is not synonymous with implementing the functionality of a graphical model in a graphical modeling and execution environment as recited in claim I. As such, Cheng fails to disclose "source code that implements functionality of said graphical model," as recited in claim 1.

Examiner's response:

b) Examiner disagrees. Cheng et al. clearly disclose source code that implements functionality of said graphical model *(see Figure 4; Paragraph [0023], "FIG. 4 shows an exemplary command graphical user interface ("GUI") 200 for command structure manifest 110 described with respect to FIG. 2. Command structure manifest 110 enables a developer to* *visually manipulate the command structure by adding and deleting command nodes at any level." and "... GUT (sic) 200 also shows parameters and handler functions associated with each command node.").* Note that the handler function code implements each command node (functionality) of the command structure (graphical model).

Conclusion

14. The prior art made of record and not relied upon is considered pertinent to Applicant's disclosure.

Any inquiry concerning this communication or earlier communications from the Examiner should be directed to Qing Chen whose telephone number is 571-270-1071. The Examiner can normally be reached on Monday through Thursday from 7:30 AM to 4:00 PM. The Examiner can also be reached on alternate Fridays.

If attempts to reach the Examiner by telephone are unsuccessful, the Examiner's supervisor, Wei Zhen, can be reached on 571-272-3708. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Any inquiry of a general nature or relating to the status of this application or proceeding should be directed to the TC 2100 Group receptionist whose telephone number is 571-272-2100.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

QC / ac
August 13, 2007

WEI ZHEN
SUPERVISORY PATENT EXAMINER